Proceedings of the Society for Computation in Linguistics

Volume 2 Article 22

2019

C-Command Dependencies as TSL String Constraints

Thomas Graf
Stony Brook University, mail@thomasgraf.net

Nazila Shafiei
Stony Brook University, nazila.shafiei@stonybrook.edu

C-command dependencies as TSL string constraints

Thomas Graf
Stony Brook University
mail@thomasgraf.net

Nazila Shafiei
Stony Brook University
nazila.shafiei@stonybrook.edu

Abstract

We provide a general formal framework for analyzing c-command based dependencies in syntax, e.g. binding and NPI licensing, from a subregular perspective. C-command relations are represented as strings computed from Minimalist derivation trees, and syntactic dependencies are shown to be input-output tier-based strictly local over such strings. The complexity of many syntactic phenomena thus is comparable to dependencies in phonology and morphology.

1 Introduction

The last decade has seen tremendous progress in the subregular analysis of phonology and morphology (see Chandlee 2017, Heinz 2018 and references therein). The subregular program is concerned with identifying proper subclasses of the regular languages that are sufficiently powerful for natural language, and to convert these tighter complexity bounds into typological predictions and new learning algorithms.

Extending the subregular approach to syntax has proven more challenging because syntactic dependencies are not regular over strings, let alone subregular (Chomsky, 1957; Huybregts, 1984; Shieber, 1985; Michaelis and Kracht, 1997; Kobele, 2006). But syntactic dependencies seem to be subregular over tree structures (Graf, 2012). More recently, a series of arguments (Graf, 2018a,b; Graf et al., 2018) has been presented that dependencies in phonology, morphology, and syntax are all of comparable subregular complexity once one picks a suitable data structure (string-like vs. trees). More precisely, all dependencies are conjectured to belong to natural extensions of the class *tier-based strictly local* (TSL; Heinz et al., 2011; Graf and Mayer, 2018).

While this claim is well-supported for phonology and morphology, it is on shakier ground for syntax. It seems to hold for subcategorization and displacement (Merge and Move in Chomskyan terms) and some kinds of NPI-licensing (Vu, 2018). But it may fail for constraints based on c-command, which are numerous in syntax. This paper argues that these dependencies are also TSL given the right kind of representation.

We define a general procedure for converting c-command dependencies to string languages. Intuitively, our procedure amounts to little more than making c-command a basic relation in syntactic representations (cf. Frank and Vijay-Shanker, 1999). Formally, we represent Minimalist grammar derivations (Stabler, 1997) as dependency trees (cf. Brody, 2000; Kobele, 2002) and then associate each node with a string of nodes that c-command it. Principle A of binding theory, for example, then requires that a string that ends in a reflexive must contain a potential binder somewhere between the reflexive and the head of its binding domain. Based on a preliminary analysis of c-command dependencies in the literature, we contend that they all fit into extensions of TSL that have been proposed for phonology and morphology.

The paper proceeds as follows. We first discuss some general preliminaries (§2), including MGs, dependency trees, and subregular syntax. After that, c-strings are introduced as a string-based representation of c-command relations (§3.1), and various syntactic phenomena are characterized in these terms (§3.2). We then show that the dependencies of these phenomena are input-output tier-based strictly local (Graf and Mayer, 2018) over c-strings (§4), and we explain why this likely holds for all c-command dependencies in syntax. Some final complications introduced by movement are discussed in §5.

Proceedings of the Society for Computation in Linguistics (SCiL) 2019, pages 205-215.
New York City, New York, January 3-6, 2019

2 Preliminaries

While the ideas presented in this paper are fairly intuitive, a sufficiently rigorous description requires a broad formal background. We first introduce some basic notation (§2.1) and then discuss Minimalist grammars as a formal model of syntax (§2.2). We also explain how their derivation can be encoded as dependency trees, which will be used in §3 to easily compute the c-commanders for any given node. After that, §2.3 surveys some recent developments on subregular syntax and why this paper pursues a different route.

2.1 Formal language theory

We follow the standard notation for strings: Σ by default denotes a fixed alphabet, ε is the empty string, S^* is the Kleene closure of set S, S^+ is $S^* - \{\varepsilon\}$, $\overline{S}$ is $\Sigma - S$, and instead of $\{a\}$ we may simply write a.

2.2 Minimalist grammars

In order to assess the complexity of syntactic dependencies, we need a formal model of syntax. We choose Minimalist grammars (MGs; Stabler, 1997, 2011). MGs are inspired by Chomsky (1995) but can incorporate a wide range of ideas from the syntactic literature (see Graf 2013 and references therein). This makes them an ideal testing ground for c-command dependencies.

Every MG is a finite set of lexical items that are annotated with strings of features to drive the structure-building operations Merge and Move.

Example 1 (Merge)**.** The noun *car* only carries the *category feature* N^-. The determiner *the* carries a matching *selector feature* N^+. The full feature string for *the* is $\mathrm{N}^+\mathrm{D}^-$. This indicates that *the* first selects a noun and then acts as DP that can be selected by anything with the matching selector feature D^+. The possessive marker *'s*, on the other hand, must select both a noun as its complement and a DP as its specifier. Hence its feature string is $\mathrm{N}^+\mathrm{D}^+\mathrm{D}^-$.

Example 2 (Move)**.** The determiner *which* is similar to *the*, except that it also undergoes wh-movement after it has been selected. Therefore its feature string must include a *licensee feature* wh^- at the very end: $\mathrm{N}^+\mathrm{D}^-\mathrm{wh}^-$. At least in some cases the landing site of the wh-mover is an empty CP-specifier. In order to allow for this, the grammar must contain a lexical item whose phonetic exponent is ε and whose feature string is $\mathrm{T}^+\mathrm{wh}^+\mathrm{C}^-$. Here wh^+ is a *licensor feature* that marks the empty C-head as a landing site for wh-movement.

As every MG builds a phrase structure tree via a sequence of Merge and Move steps, every syntactic tree can be fully specified by its derivation. The derivation, in turn, can be represented as a tree, which may take the form of a dependency tree as shown in Fig. 1 (cf. Brody, 2000; Kobele, 2002). In an MG dependency tree, a node d is the daughter of node n iff n selects d. For MGs, this means that X^- on d and a matching X^+ of n were checked as part of the same Merge operation. We assume that the linear order of siblings is the reverse of the order of selection, so that the first argument of a head is its rightmost daughter. Movement is not indicated in dependency trees as it can be inferred from the distribution of licensor and licensee features.

2.3 MGs as subregular tree languages

It has been known for a long time that every MG's set of well-formed derivation trees forms a regular tree language (Michaelis, 2001; Kobele et al., 2007). While these results build on a different derivation tree format, they also hold for the dependency tree format in this paper because the latter can be obtained from the former by a regularity-preserving tree transduction.

It has also become clear, though, that MG derivation tree languages are not just regular, but subregular (Graf, 2012). Most recently, Graf (2018b) showed that they belong to the formal class TSL. This class was first defined for strings to handle some phenomena in phonology (Heinz et al., 2011). Graf lifts TSL from strings to trees. Intuitively, TSL masks out parts of the tree that are irrelevant for a given dependency and then puts locally bounded constraints on the remainder. Graf limits his attention to Merge and Move, but follow-up work has since argued that NPI-licensing and morphological case also fit into this class (Vu, 2018; Vu et al., 2019). If correct, this would not only suggest a new kind of computational parallelism between language modules, but might also prove useful for parsing and learning algorithms.

Although we believe this work to be very promising, it is still based on detailed case studies and has not yielded any general properties that guarantee that TSL is a safe upper complexity

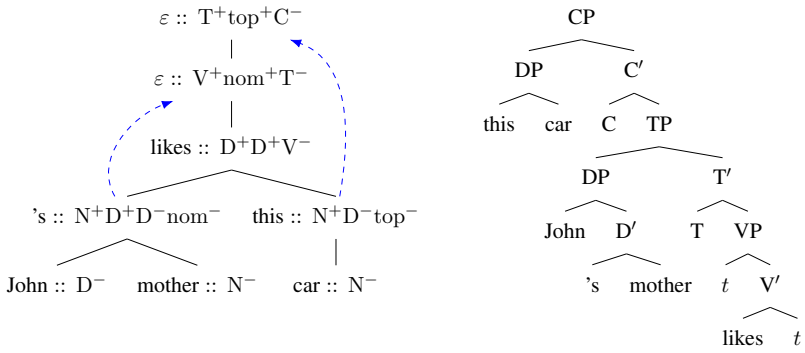

Figure 1: A dependency tree representation of an MG derivation, with its corresponding phrase structure tree to the right. Movement is indicated by arrows for the reader's convenience, it is not part of the tree. Crucially, moving phrases remain in their base position in the dependency tree.

bound for all attested syntactic dependencies. In addition, the step from strings to trees complicates the comparison to phonology and morphology. Several subregular classes besides TSL have been argued to play a central role in these domains, e.g. SP (Rogers et al., 2010) and IBSP (Graf, 2017, 2018a). The latter is difficult to lift from strings to trees, whereas the former seems ill-suited for syntactic dependencies based on preliminary work of ours. For these reasons, a string-based view of syntactic dependencies is a welcome alternative that makes it easier to compare phonology and syntax. As we argue in this paper (§4.2), it also provides some general insights that make it very likely that all c-command dependencies fall within a subregular region that is also occupied by dependencies in phonology and morphology.

3 C-command relations as strings

We are finally in a position to demonstrate how c-command dependencies reduce to constraints on strings. We first present the general idea in §3.1 and then apply it to several examples in §3.2. We only describe the relevant string languages and wait until §4 to show that they fit into (extensions of) TSL.

3.1 Idea and definition

The dependency graphs make it very easy to enforce c-command requirements for any given node — at least if one puts aside movement, as we will do until §5. Until then, "c-command" only means "c-command between base positions before movement".

If u is a left sibling of v in an MG dependency tree t, then (the phrase headed by) u c-commands (the phrase headed by) v. This is illustrated by *John* and *mother* in Fig. 1. In addition, v is c-commanded by almost every node that dominates it. Looking again at Fig. 1, we see that the T-head and the C-head dominate *John* in the dependency tree and c-command *John* in the phrase structure tree (at the position marked by the trace in Spec,VP).

One minor wrinkle is the relation between heads and their specifiers. The former do not c-command the latter, but they do so according to our definition. Hence our notion of command is actually a hybrid of c-command and m-command (Aoun and Sportiche, 1983). One may call it *d[erivational]-command* to emphasize this difference: a node d-commands all its right siblings, all its daughters, and everything that is d-commanded by its siblings or daughters. As far as we can tell, the minor differences between c-command and d-command are immaterial for all syntactic dependencies that have been argued to involve c-command, and we will continue to use c-command as the general term instead of d-command whenever this does not cause any confusion.

Note that d-command is a highly natural relation over derivation trees as it encodes prominence with respect to feature checking. Based on the definition above, x d-commands y iff x had one of its Merge features checked more recently than y. A specifier s of head h d-commands a complement c of h because s has been selected more recently than c. And h d-commands s because h is selected by some other head after s has been selected. C-command as a basic relation of syntax is difficult to motivate (but see Epstein et al. 1998 and Chametzky 2000, 2011). D-command, on the

other hand, falls out naturally from the MG feature calculus and thus is a plausible primitive for syntactic dependencies.[1]

One can automatically calculate for every node a string representation of the relevant command relations. Let T be a tree such that node m has the daughters $d_1, \ldots, d_i, d, d_{i+1}, \ldots, d_n$, with $n \geq 0$. The *immediate c[ommand]-string* $ics(d)$ of d is the string $d\ d_i \cdots d_1$. For every node n of T, its *c[ommand]-string* $cs(n)$ is recursively defined as shown below, where $\cdot$ indicates string concatenation:[2]

$$cs(n) := \begin{cases} ics(n) & \text{if } n \text{ is the root of } T \\ ics(n) \cdot cs(m) & \text{if } m \text{ is } n\text{'s mother} \end{cases}$$

Note that c-strings sort d-commanders in a bottom-up fashion, which usually yields the reverse of their linear order in the string. For improved readability, we omit all features and replace empty heads by their category.

Example 3. Consider once more the dependency graph in Fig. 1. While the immediate command string of *car* is just *car*, its c-string is *car this 's likes T C*.

The next subsection illustrates how a number of syntactic phenomena can be reinterpreted as constraints on c-strings. Even more strikingly, these constraints are fairly intuitive, and as we will see in §4, their subregular complexity is largely in line with what has been found in phonology and morphology.

3.2 Examples of syntactic constraints

It is beyond the purview of this paper to exhaustively analyze every syntactic dependency that has been claimed to involve c-command. Consequently, we limit ourselves to a representative sample of NPI-licensing (§3.2.1) and binding effects (§3.2.2–§3.2.4). Due to space restrictions, we are also limited to high-level discussions that illustrate the general ideas but do not account for all details and potential complications (cf. §3.2.5).

[1] If one wants to include c-command relations created by movement, it suffices to broaden the definition such that any kind of feature checking induces d-command relations, not just those affecting category and selector features.

[2] C-strings over dependency trees are closely related to the concept of spine languages (cf. Martens et al., 2008; Graf, 2012). In spine languages, $ics(n)$ and $cs(m)$ would be separated by a distinguished symbol, e.g. ♣. We omit this here for simplicity, but point out that preliminary work of ours suggests that ♣ is needed in some cases to correctly define locality domains.

3.2.1 NPI licensing

NPIs can occur in a variety of contexts, but the central syntactic licensing condition is c-command by a downward-entailing operator. We can capture this in terms of conditions on string languages. If a c-string ends in an NPI, then it must also contain a licensor such as *no* or *nobody*. So the required string language is *NPI* $\cdots$ {*no, nobody*} $\cdots$. This ensures that no NPI can ever occur without a c-commanding licensor.

Example 4. Consider the ill-formed **Every student said that the train ever arrives on time*. Assuming that *ever* is a functional head between V and T, its c-string is *ever T that every said T C*. Since this string is not a member of the NPI-licensing language, *ever* is not licensed and the sentence is illicit.

As is witnessed by *Have you ever been to Russia*, NPIs are also licensed in questions. We can readily accommodate this by adding interrogative C-heads to the set of licensors. Overall, then, the c-string pattern for NPI-licensing is *NPI* $\cdots$ *Lic* $\cdots$, where *Lic* is a downward entailing quantifier or an interrogative C-head.

3.2.2 Locally bound reflexives

Principle A requires that reflexives must be bound within their binding domain, usually a TP containing an abstract subject. When construed as a distributional constraint, this reduces to the presence of a DP in the binding domain that c-commands the reflexive and matches its ϕ-features (person, number, and gender).

In MGs, every TP with an abstract subject is headed by a lexical item with category feature T^- and licensor feature nom^+. Let T be the shorthand for any such LI, $R[\phi]$ a reflexive, and $D[\phi]$ a matching determiner. Since MGs have finite lexicons, one can list for each reflexive which determiners match its ϕ-features. Then Principle A corresponds to the string language $R[\phi]\ \overline{T}^* D[\phi] \cdots$, where $\overline{T}^*$ is a (possibly empty) string that contains no instances of T.

Example 5. Contrast the well-formed *John said that Mary likes herself* with the ill-formed **John said that Mary likes himself*. The former has the c-string *herself Mary likes T that John said T C*. Since *Mary* is ϕ-feature compatible with *herself* and occurs between the reflexive and the T-head, Principle A is satisfied. In the ill-formed sentence, *herself* is replaced by *himself*. The closest com-

patible D-head is *John*, but since a finite T-head occurs between *John* and the reflexive, Principle A is violated.

3.2.3 Non-locally bound reflexives

The reflexive *sig* in Swedish only partially obeys Principle A. It still requires a syntactic binder, but said binder must not be part of the same binding domain (Kiparsky, 2002). The corresponding c-string language is $R[\phi] \cdots T \cdots D[\phi] \cdots$.

Example 6. Suppose for the sake of argument that English *himself* behaves like Swedish *sig*, and consider again *John said that Mary likes himself/ herself*. Then the sentence with *herself* would be ill-formed, but not the one with *himself*. And as the reader may verify for themselves, only the c-string of *himself* fits the pattern above.

3.2.4 Principle B

Principle B regulates the distribution of syntactically bound pronouns. Pronouns in English can always be discourse-bound, so their distribution is unproblematic. Marathi, on the other hand, has pronouns that must be syntactically bound (Kiparsky, 2002). Like standard pronouns, they I) must be non-locally bound, and II) cannot have the same referent as a local c-commander. Condition II) makes the distribution of Marathi pronouns much harder than that of Swedish *sig*-reflexives.

From a distributional perspective, Principle B enforces a counting condition: if a pronoun is c-commanded by n pronouns in the same binding domain, then at least n potential antecedents must c-command the binding domain in order to furnish at least one grammatical reading. Temporarily putting aside ϕ-features, this yields the string template $\alpha T \beta$ such that $\alpha, \beta \in \Sigma^*$, α starts with a pronoun, and the total number of R-expressions (i.e. D-heads) in β exceeds the number of pronouns in α; pronouns are counted as R-expressions in β because they can bind pronouns in α. Without further assumptions, this pattern is context-free and thus much more complicated than any of the previous c-string languages.

But Graf and Abner (2012) show that even though a binding domain may contain an unbounded number of pronouns, most of them can share the same referent. Pronouns within distinct adjuncts can pick out the same referent, and other constructions such as coordination are very limited in how they interact with bound pronouns. As a result, there is an upper bound n on the number of distinct antecedents that are needed to license all pronouns in a binding domain. This guarantees regularity and, as we will see in §4.3, subregularity by virtue of being definable in IO-TSL.

Example 7. Suppose for the sake of argument that English pronouns always require syntactic antecedents. Then the sentence *John thinks that he likes him* is ungrammatical, whereas *John told Bill that he likes him* is well-formed. This immediately follows from the c-string of the second pronoun in each sentence. One is *him he likes T that John thinks T C* and thus does not furnish enough antecedents for *him* and *he*. The other c-string provides the required minimum of two antecedents, namely *John* and *Bill*: *him he likes T that Bill John told T C*.

3.2.5 Caveats

Each one of the phenomena above has several complications. What counts as an NPI licensor depends on the position of the NPI. For example, *every* can license an NPI if it occurs in a relative clause modifying the argument of *every*. Similarly, many reflexives are exempt from Principle A when they occur inside certain adjuncts, and some languages like Icelandic allow for long-distance binding only under specific circumstances. These are serious complications. A careful analysis of any one of these phenomena arguably requires a full-length journal paper. Nonetheless we believe that these complications would render the string patterns more complicated, but not more complex. The reason for this is explained at the end of the next section, which analyzes the subregular complexity of syntactic dependencies over c-strings.

4 Subregular complexity

We have successfully reduced c-command dependencies to regular string languages. But regularity is a loose upper bound. As we explain next, the c-string languages from the previous section all fit into IO-TSL, the class of *input-output tier-based strictly local* string languages. This class has recently been proposed as an upper bound on phonotactic complexity (Graf and Mayer, 2018). We first present the defintion IO-TSL (§4.1) and provide a general IO-TSL strategy for capturing dependencies over c-strings (§4.2). This strategy is then exemplified with IO-TSL grammars for each one of the templates from the previous section (§4.3). In §4.4, we explain why IO-TSL is a meaningful

upper bound that makes testable predictions about the shape of syntactic dependencies.

4.1 Defining IO-TSL

IO-TSL is an extension of the *strictly local* (SL) languages. A language is SL iff it can be described by a list of finitely many forbidden substrings. IO-TSL enhances this with a local tier projection mechanism: whether a symbol is projected on a tier depends on the symbol itself, its local context in the string (i.e. up to m symbols before and/or after it), and up to n previous symbols on the tier.

Example 8. Unbounded tone plateauing forbids any instances of the low tone L to occur between two high tones H. So HLLL and LLLH are well-formed, but HLLLH is not. One can capture this by projecting L iff it occurs immediately to the right of H, and projecting H iff the previous symbol on the tier is L. The strings above would get the tiers L, empty, and LH, respectively. If one forbids the substring LH on tiers, HLLLH is ruled out as desired while HLLL and LLLH are still generated.

The rest of this subsection formalizes this intuitive idea. Let Σ be some fixed alphabet and $s \in \Sigma^*$. The set $f_k(s)$ of *k-factors* of s consists of all the length-k substrings of $\rtimes^{k-1} s \ltimes^{k-1}$, where $\rtimes, \ltimes \notin \Sigma$ and $k \geq 1$.

Definition 1. A stringset $L \subseteq \Sigma^*$ is *strictly k-local* (SL-k) iff there is some $G \subseteq (\Sigma \cup \{\rtimes, \ltimes\})^k$ such that $L = \{s \in \Sigma^* \mid f_k(s) \cap G = \emptyset\}$.

Intuitively, G defines a *grammar* of forbidden substrings that no well-formed string may contain. The class SL of strictly local stringsets is $\bigcup_{k \geq 0}$ SL-k.

Example 9. The string language $(ab)^+$ is generated by the grammar $G := \{\rtimes\ltimes, \rtimes b, aa, bb, a\ltimes\}$ and thus is SL-2. For instance, aba is illicit because $f_2(aba) \cap G = \{a\ltimes\} \neq \emptyset$, whereas $f_2(abab) \cap G = \emptyset$.

Example 10. The finite language of all strings of length 3 is SL-4. It is generated by the grammar $\{abcd \mid a, b, c, d \in \Sigma\} \cup \{abcd \mid a = \rtimes, d = \ltimes, b, c \in \Sigma \cup \{\rtimes, \ltimes\}\}$.

The tier projection mechanism of IO-TSL is defined in terms of contexts that specify when a given symbol should be added to the tier. An (i, j)-*context* c is a 4-tuple $\langle \sigma, b, a, t \rangle$ with $\sigma \in \Sigma$, t a string over $\Sigma \cup \{\rtimes\}$ of length $j - 1$, and a and b strings over $\Sigma \cup \{\rtimes, \ltimes\}$ of combined length $i - 1$. The basic idea is that c specifies that σ should be projected whenever both of the following hold: it occurs between the substrings b (look-*b*ack) and a (look-*a*head), and the tier constructed so far ends in t. Given a set of contexts $c_1, c_2, \ldots, c_n$, we call it an (i, j)-*context set* iff for every c_m ($1 \leq m \leq n$) there are $i_m \leq i$ and $j_m \leq j$ such that c_m is an (i_m, j_m)-context.

Definition 2. Let C be an (i, j)-context set. Then the *input-output strictly (i, j)-local* (IOSL-(i, j)) tier projection π_C maps every $s \in \Sigma^*$ to $\pi'_C(\rtimes^i, s\ltimes^i, \rtimes^j)$, where for $\sigma \in \Sigma$ and $a, b, t, u, v, w \in (\Sigma \cup \{\rtimes, \ltimes\})^*$, it holds that $\pi'_C(ub, \sigma av, wt)$ is

$$\begin{array}{ll} \varepsilon & \text{if } \sigma av = \varepsilon, \\ \sigma \pi'_C(ub\sigma, av, wt\sigma) & \text{if } \langle \sigma, b, a, t \rangle \in C, \\ \pi'_C(ub\sigma, av, wt) & \text{otherwise.} \end{array}$$

Example 11. Let $\Sigma := \{a, b, c\}$ and consider the tier projection that always projects the first and last symbol of the string, always projects a, never projects c, and projects b only if the previous symbol on the tier is a. This projection is IOSL-(2,2). The context set contains all the contexts below, and only those:

- $\langle \sigma, \rtimes, \varepsilon, \varepsilon \rangle$ for all $\sigma \in \Sigma$,
- $\langle \sigma, \varepsilon, \ltimes, \varepsilon \rangle$ for all $\sigma \in \Sigma$,
- $\langle a, \varepsilon, \varepsilon, \varepsilon \rangle$,
- $\langle b, \varepsilon, \varepsilon, a \rangle$.

Definition 3. A stringset $L \subseteq \Sigma^*$ is *input-output tier-based strictly (i, j, k)-local* (IO-TSL-(i, j, k)) iff there exists an IOSL-(i, j) tier projection π_C and an SL-k language K such that $L := \{s \in \Sigma^* \mid \pi_C(s) \in K\}$. It is IO-TSL iff it is IO-TSL-(i, j, k) for some i, j, and k.

The class TSL-k defined in Heinz et al. (2011) is identical to IO-TSL-$(1, 1, k)$. This shows that IO-TSL is indeed a generalization of TSL. In fact, I-TSL and O-TSL have been independently proposed in computational phonology (Baek, 2017; De Santo and Graf, 2017; Mayer and Major, 2018; Yang, 2018), and Graf and Mayer (2018) show that the combination of the two into IO-TSL furnishes the additional power that is required for Sanskrit *n*-retroflexion. Under the assumption that

dependencies in phonology and syntax are of comparable complexity, IO-TSL is a natural candidate for a tighter upper bound on the complexity of constraints on c-strings.

4.2 General IO-TSL strategy

The dependencies in §3.2 — and syntactic dependencies in general — all share the common property that even though there is no upper bound on the length of c-strings, only a finitely bounded number of elements actually matter. Which elements matter depends only on the local context and which symbols are already on the tier. This allows for a very general IO-TSL strategy: construct tiers such that they only contain the relevant elements. Then there are only finitely many distinct tier configurations, and hence the set of well-formed tiers is finite. As every finite string language is SL-k for some k (cf. Ex. 10), separating the well-formed tiers from the ill-formed ones becomes trivial. So the central challenge for an IO-TSL treatment of c-command dependencies lies in the construction of tiers, not the constraints on those tiers.

The tier construction process also follows a general template. We project tiers from left to right, and the very first element is always projected — this is an input-sensitive projection step. We then use an output-sensitive projection strategy to only project relevant elements.

Example 12. Suppose the first symbol to be projected is an NPI. Then only an NPI-licensor can be projected next. After the first NPI-licensor has been projected, no more symbols are put on the tier. Since a single NPI-licensor is enough, we never project more than one. So the tier either has the form *NPI NPI-licensor* or just *NPI*. The former is well-formed, the latter is not.

Example 13. If the first element is a reflexive that is subject to Principle A, only T-heads and matching D-heads need to be projected. Again nothing is projected after this second projection step as Principle A is already satisfied or violated depending on whether the second symbol is T or $D[\phi]$. The tier then has one of the following three forms: *reflexive D*, *reflexive T*, or just *reflexive*. The first one is well-formed, the other two ill-formed.

If the first symbol on the tier is not subject to any constraints, then nothing else is projected. Proceeding in this fashion, we can define a single IO-TSL grammar that generates every well-formed c-string while forbidding those that violate one of our string patterns from §3. This shows that the whole system of syntactic dependencies is IO-TSL, not just each individual dependency.

4.3 Example grammars for dependencies

Let us now apply this idea to the c-string templates from §3, repeated here for the readers convenience:

NPI $\cdots$ *Lic* $\cdots$
$R[\phi]\ \overline{T}^*\ D[\phi] \cdots$
$R[\phi] \cdots T \cdots D[\phi] \cdots$
$\alpha T \beta$ (α starts with pronoun,
β has more R-expressions than α has pronouns)

Now consider the corresponding projection contexts for any arbitrary MG G. We first include $\langle \sigma, \varepsilon, \ltimes, \varepsilon \rangle$ for all $\sigma \in G$ so that the first symbol is always projected. Next we add $\langle l, \varepsilon, \varepsilon, t \rangle$ for all $l \in G$ and $t \in G^*$ such that:

NPI t is an NPI and l an NPI licensor, or

Reflexive t is a reflexive, and l is either a T-head carrying nom^+ or a D-head with matching ϕ-features, or

sig either t is a *sig*-reflexive and l is a T-head carrying nom^+, or $t = uv$ with u a *sig*-reflexive and v a T-head carrying nom^+ and l is a D-head with matching ϕ-features.

The definition for **sig** is convoluted because it accounts for both projection of T after *sig* and projection of $D[\phi]$ after T and *sig*.

Next we forbid all bigrams of the form lp such that either l is an NPI and p is not an NPI-licensor, or l is a reflexive and p is not a matching D-head. Note that this includes cases where p is no LI at all, but rather the start/end of a tier. We also forbid all trigrams that start with a *sig*-reflexive but do not continue with a suitable T-head and a matching D-head.

Example 14. Consider once more the illicit NPI in example 4. Given the c-string *ever T that every said T C*, the constructed tier is just *ever*, which is forbidden. If *every* is replaced with *no*, the tier becomes *ever no* and thus is well-formed. If *ever* were *always* instead, the tier would just be *always*, which is never illicit because no special licensing is needed. Note that the tier never grows past *ever no*, even for sentences like *No student told no professor that no train ever arrives on time*.

Example 15. Suppose as in example 6 that English reflexives behaved like Swedish *sig* and thus require a non-local binder. The c-string *herself Mary likes T that John said T C* results in the illicit tier *herself T*. But if *herself* is replaced by *himself*, one obtains the well-formed tier *himself T John* instead. This is also the tier for much longer sentences such as *Bill thinks Peter doubts that John said that Mary likes him*.

For Principle B, the projection strategy is very similar.

- Project the first element, which is a pronoun that must be syntactically bound.
- Also project such pronouns if the tier so far contains i such pronouns and nothing else, where i is less than some fixed bound n.
- Project a T-head carrying nom^+ if the tier only contains pronouns so far.
- Project an R-expression if the tier already contains a T-head and at most $n-1$ R-expressions.

With this projection strategy, the longest possible tier is of the form $pro^n\ T\ R^n$, and all illicit tiers can be filtered out by $(2n+1)$-grams.

As pointed out earlier, this analysis ignores complicating factors such as additional licensing configurations for NPIs, or ϕ-feature matching in Principle B. But this does not affect the core property that guarantees the IO-TSL nature of c-command dependencies: Each dependency requires only a finite number of elements on the tier, and by considering the entire tier built so far one can ensure that no extraneous material is projected. In addition, each LI can only be subject to finitely many licensing conditions. These two facts jointly entail that the length of tiers can always be finitely bounded, which makes it trivial to provide an SL grammar that rules out all illicit tiers. It is because of this fixed bound on the number of elements that matter for any given c-command dependency that IO-TSL is a safe upper bound on the complexity of syntactic dependencies over c-strings. This is also the reason why we contend that IO-TSL could accommodate empirically more adequate characterizations of the phenomena in §3 — the number of relevant lexical items would still be finitely bounded within any given c-string.

Admittedly, this strategy comes at the potential cost of large contexts. Fairly small contexts seem to suffice for realistic examples, though, and a smarter, less brute-force strategy may be able to reduce their size even further. We would not be surprised if most c-command dependencies turn out to belong to IO-TSL-$(3,3,3)$ or perhaps even IO-TSL-$(2,2,2)$.

4.4 Limits of IO-TSL dependencies

IO-TSL makes some strong predictions about what shape linguistic dependencies can take and how multiple c-command conditions may be interwoven.

Example 16. Consider an unattested variant of the long-distance Principle A that applies to Swedish *sig*. In this variant, the reflexive and the antecedent not only have to be separated by a T-head, but each intervening T-head must be c-commanded by some functional head F that is lower than the next higher T-head. So the c-string language is not just $\cdots D[\phi] \cdots T \cdots R[\phi]$, but rather $\alpha R[\phi]$ such that I) α contains at least one $D[\phi]$ and, II) F occurs between every two instances of T.

This string language is not IO-TSL because we can no longer omit projecting all T-heads to the tier. To see this, contrast the well-formed pattern $D[\phi](F\ T)^*\ F\ T\ (F\ T)^*R[\phi]$ against the ill-formed $D[\phi](F\ T)^*\ T\ (F\ T)^*R[\phi]$. The only difference is the unlicensed T-head in the middle, so this T-head must end up on the tier in order to distinguish well-formed from ill-formed strings. But there is no context that can uniquely identify just this T-head without projecting other T-heads or F-heads. The set of well-formed tiers then would be an infinite subset of $\{T,F\}^*\ D[\phi]\ \{T,F\}^*\ R[\phi]$, which is not SL unless the distance between $D[\phi]$ and $R[\phi]$ is finitely bounded. Since there are no additional factors that guarantee such a bound on the distance between reflexive and licensor, our unattested variant of *sig*-licensing is not IO-TSL.

This example highlights a crucial limitation of IO-TSL: a long-distance dependency cannot apply across an unbounded number of long-distance dependencies that interact with it. The F-licensing of T-heads would be unproblematic if it were an independent constraint that must always be satisfied, rather than just being an extra condition on *sig*-licensing. For then it would be a condition on all c-strings that start with T-heads and could be omitted in c-strings that start with reflexives. Al-

ternatively, F-licensing of T-heads could be captured if the distance between F and T were locally bounded. In this case, one could project F and T only once and skip all other locally licensed T-heads. Similarly, it would suffice to project at most one unlicensed T-head. The result would be tiers of the form $D[\phi]\alpha R[\phi]$ where α is $T\ F\ T$ or $F\ T\ T$. Then the tier language would once again be finite. So a long-distance dependency can interact with other dependencies, but either the number of those dependencies or their locus of application must be finitely bounded.

It is worth mentioning that this unattested example pattern is easily defined in first-order logic, which so far has been the only safe upper bound on syntactic dependencies (Graf, 2012). The IO-TSL perspective of c-command dependencies thus improves on previous work by ruling out some linguistically undesirable patterns.

5 Adding movement

The discussion so far has largely ignored the effects of movement on c-command relations. Under the standard raising analysis, *no* does not c-command *ever* from its base position in *[No train]$_i$ has ever seemed to me t_i to t_i arrive on time*. The c-command relation is derived by movement. Our current notion of c-strings fails to capture this.

For most dependencies discussed in this paper, it only matters that a dependent element has at least one licensor in its c-string. In this case, movement effects can be accommodated by modifying the construction of c-strings such that a mover is also copied into its landing sites. The c-string for *ever* in the previous example would be *ever no has C* after the mover *no* is copied into the position corresponding to its landing site.

But this covers only movement of a licensor. Sometimes, a dependent element is licensed only because it has moved out of a locality domain into a higher position where it is accessible to its licensor. In such cases, the c-string would be truncated by deleting all material between the licensor and the relevant occurrence, effectively pushing the licensee into a higher position.

This strategy shifts a lot of the burden to the correct construction of c-strings, which may be particularly complicated when both the licensor and the licensee move. The construction of the appropriate c-string from a dependency tree is still definable in first-order logic and hence subregular, but this is a very generous upper bound. An in-depth exploration of movement from a subregular perspective has to be left for future work.

Conclusion

We have defined a string-based representation format over dependency trees that allows for c-command dependencies to be easily evaluated for any given node. The dependencies over these strings all fall within the class IO-TSL, which was first defined for phonology (Graf and Mayer, 2018). This paper marks but a first step towards a subregular theory of syntactic dependencies, and a lot remains to be done.

The current approach only measures the complexity of a syntactic dependency with respect to a specific node. To check the whole dependency tree, one has to evaluate the c-string of each node. We do not know whether the TSL-approach of Graf (2018b) provides a method for doing so. As long as all dependencies are IO-TSL, though, the well-formedness of the whole dependency tree can be verified by a deterministic top-down tree automaton with a look-ahead of 1. This implies subregular complexity and may even allow for highly efficient parsing algorithms.

Dependencies that go beyond c-command cannot be handled with our approach. This includes binding via sub-command, parasitic gaps, and across-the-board movement as an exception to the coordinate structure constraint. It remains to be seen whether they can be accommodated with tree tiers as proposed in Graf (2018b), or whether a completely new perspective is needed for these phenomena.

References

Joseph Aoun and Dominique Sportiche. 1983. On the formal theory of government. *Linguistic Review*, 2:211–236.

Hyunah Baek. 2017. Computational representation of unbounded stress: Tiers with structural features. Ms., Stony Brook University; to appear in *Proceedings of CLS 53*.

Michael Brody. 2000. Mirror theory: Syntactic representation in perfect syntax. *Linguistic Inquiry*, 31:29–56.

Robert A. Chametzky. 2000. *Phrase Structure: From GB to Minimalism*. Blackwell, Oxford.

Robert A. Chametzky. 2011. No derivation without representation. In Cedric Boeckx, editor, *The Oxford Handbook of Linguistic Minimalism*, pages 311–326. Oxford University Press, Oxford.

Jane Chandlee. 2017. Computational locality in morphological maps. *Morphology*, 27:599–641.

Noam Chomsky. 1957. *Syntactic Structures*. Mouton, The Hague.

Noam Chomsky. 1995. *The Minimalist Program*. MIT Press, Cambridge, MA.

Aniello De Santo and Thomas Graf. 2017. Structure sensitive tier projection: Applications and formal properties. Ms., Stony Brook University.

Samuel D. Epstein, Erich M. Groat, Ruriko Kawashima, and Hisatsugu Kitahara. 1998. *A Derivational Approach to Syntactic Relations*. Oxford University Press, Oxford.

Robert Frank and K. Vijay-Shanker. 1999. Primitive c-command. Ms., John Hopkins University and University of Delaware.

Thomas Graf. 2012. Locality and the complexity of Minimalist derivation tree languages. In *Formal Grammar 2010/2011*, volume 7395 of *Lecture Notes in Computer Science*, pages 208–227, Heidelberg. Springer.

Thomas Graf. 2013. *Local and Transderivational Constraints in Syntax and Semantics*. Ph.D. thesis, UCLA.

Thomas Graf. 2017. The power of locality domains in phonology. *Phonology*, 34:385–405.

Thomas Graf. 2018a. Locality domains and phonological c-command over strings. To appear in *Proceedings of NELS 2017*.

Thomas Graf. 2018b. Why movement comes for free once you have adjunction. To appear in *Proceedings of CLS 53*.

Thomas Graf and Natasha Abner. 2012. Is syntactic binding rational? In *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11)*, pages 189–197.

Thomas Graf, Alëna Aksënova, Hyunah Baek, Aniello De Santo, Hossep Dolatian, Sedigheh Moradi, Jon Rawski, Suji Yang, and Jeffrey Heinz. 2018. Tiers and relativized locality across language modules. Slides of a talk given at the 1-day workshop Parallels Between Phonology and Syntax, July 9, Meertens Instituut, Amsterdam, Netherlands.

Thomas Graf and Connor Mayer. 2018. Sanskrit n-retroflexion is input-output tier-based strictly local. To appear in *Proceedings of SIGMORPHON 2018*.

Jeffrey Heinz. 2018. The computational nature of phonological generalizations. In Larry Hyman and Frank Plank, editors, *Phonological Typology*, Phonetics and Phonology, chapter 5, pages 126–195. Mouton De Gruyter.

Jeffrey Heinz, Chetan Rawal, and Herbert G. Tanner. 2011. Tier-based strictly local constraints in phonology. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 58–64.

M. A. C. Huybregts. 1984. The weak adequacy of context-free phrase structure grammar. In Ger J. de Haan, Mieke Trommelen, and Wim Zonneveld, editors, *Van Periferie naar Kern*, pages 81–99. Foris, Dordrecht.

Paul Kiparsky. 2002. Disjoint reference and the typology of pronouns. In Ingrid Kaufmann and Barbara Stiebels, editors, *More than Words*, volume 53 of *Studia Grammatica*, pages 179–226. Akademie Verlag, Berlin.

Gregory M. Kobele. 2002. Formalizing mirror theory. *Grammars*, 5:177–221.

Gregory M. Kobele. 2006. *Generating Copies: An Investigation into Structural Identity in Language and Grammar*. Ph.D. thesis, UCLA.

Gregory M. Kobele, Christian Retoré, and Sylvain Salvati. 2007. An automata-theoretic approach to Minimalism. In *Model Theoretic Syntax at 10*, pages 71–80.

Wim Martens, Frank Neven, and Thomas Schwentick. 2008. Deterministic top-down tree automata: Past, present, and future. In *Proceedings of Logic and Automata 2008*, pages 505–530.

Connor Mayer and Travis Major. 2018. A challenge for tier-based strict locality from Uyghur backness harmony. In *Proceedings of Formal Grammar 2018*. To appear.

Jens Michaelis. 2001. Transforming linear context-free rewriting systems into Minimalist grammars. *Lecture Notes in Artificial Intelligence*, 2099:228–244.

Jens Michaelis and Marcus Kracht. 1997. Semilinearity as a syntactic invariant. In *Logical Aspects of Computational Linguistics*, volume 1328 of *Lecture Notes in Artifical Intelligence*, pages 329–345. Springer.

James Rogers, Jeffrey Heinz, Gil Bailey, Matt Edlefsen, Molly Vischer, David Wellcome, and Sean Wibel. 2010. On languages piecewise testable in the strict sense. In Christan Ebert, Gerhard Jäger, and Jens Michaelis, editors, *The Mathematics of Language*, volume 6149 of *Lecture Notes in Artificial Intelligence*, pages 255–265. Springer, Heidelberg.

Stuart M. Shieber. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8(3):333–345.

Edward P. Stabler. 1997. Derivational Minimalism. In Christian Retoré, editor, *Logical Aspects of Computational Linguistics*, volume 1328 of *Lecture Notes in Computer Science*, pages 68–95. Springer, Berlin.

Edward P. Stabler. 2011. Computational perspectives on Minimalism. In Cedric Boeckx, editor, *Oxford Handbook of Linguistic Minimalism*, pages 617–643. Oxford University Press, Oxford.

Mai Ha Vu. 2018. Towards a formal description of NPI-licensing patterns. In *Proceedings of the Society for Computation in Linguistics*, volume 1, pages 154–163.

Mai Ha Vu, Nazila Shafiei, and Thomas Graf. 2019. Case assignment in TSL syntax: A case study. To appear in *Proceedings of SCiL 2019*.

Su Ji Yang. 2018. Subregular complexity in Korean phonotactics. Undergraduate honors thesis, Stony Brook University.